Probabilistic Graphical Models

Guest lecturer: David Sontag

Machine Learning and Pattern Recognition, Fall 2011
Example application: Tracking

Observe noisy measurements of missile location: $Y_1, Y_2, \ldots$

Where is the missile now?
Probabilistic approach

• Our measurements of the missile location were $Y_1, Y_2, \ldots, Y_n$

• Let $X_t$ be the *true* missile location at time $t$

• To keep this simple, suppose that the locations are discrete, i.e. $X_t$ and $Y_t$ take the values 1, ..., $k$

Grid the space:
Probabilistic approach

• First, we specify the \textit{conditional} distribution \( \Pr(X_t \mid X_{t-1}) \):

From basic physics, we can bound the distance that the missile can have traveled

• Then, we specify \( \Pr(Y_t \mid X_t) \):

With probability \( \frac{1}{2} \), \( Y_t = X_t \). Otherwise, \( Y_t \) is a uniformly chosen grid location
Probabilistic approach

• We describe the joint distribution on \(X_1, X_2, \ldots, X_n\) and \(Y_1, Y_2, \ldots, Y_n\) as follows:

\[
Pr(x_1, \ldots x_n, y_1, \ldots, y_n) = Pr(x_1) Pr(y_1 | x_1) \prod_{t=2}^{n} Pr(x_t | x_{t-1}) Pr(y_t | x_t)
\]

• To find out where the missile is now, we do marginal inference:

\[
Pr(x_n | y_1, \ldots, y_n)
\]

• To find the most likely trajectory, we do MAP (maximum a posteriori) inference:

\[
\arg \max_x Pr(x_1, \ldots, x_n | y_1, \ldots, y_n)
\]
Probabilistic graphical models

• The previous example is called a **hidden Markov model**:

\[
\Pr(x_1, \ldots, x_n, y_1, \ldots, y_n) = \Pr(x_1) \Pr(y_1 | x_1) \prod_{t=2}^{n} \Pr(x_t | x_{t-1}) \Pr(y_t | x_t)
\]

• In general, there is a 1-1 mapping between the graph structure and the factorization of the joint distribution

• Let \( V \) be the set of variables (nodes), and \( \text{pa}(i) \) denotes the parents of variable \( i \). Then,

\[
\Pr(v) = \prod_{i \in V} \Pr(v_i | v_{\text{pa}(i)})
\]

• Can infer conditional independencies from graphical model alone!
The previous example is called a **hidden Markov model**:

In general, there is a 1-1 mapping between the graph structure and the factorization of the joint distribution

Let $V$ be the set of variables (nodes), and $\text{pa}(i)$ denotes the parents of variable $i$. Then,

$$\Pr(v) = \prod_{i \in V} \Pr(v_i | v_{\text{pa}(i)})$$

Can infer conditional independencies from graphical model alone!
Probabilistic graphical models

- The previous example is called a **hidden Markov model**:

  \[
  X_1 \perp X_3 \mid X_2 \\
  \Pr(X_1, X_3 \mid X_2) = \Pr(X_1 \mid X_2) \Pr(X_3 \mid X_2)
  \]

- In general, there is a 1-1 mapping between the graph structure and the factorization of the joint distribution

- Let \( V \) be the set of variables (nodes), and \( \text{pa}(i) \) denotes the parents of variable \( i \). Then,

  \[
  \Pr(v) = \prod_{i \in V} \Pr(v_i \mid \text{v}_{pa(i)}) \quad \text{Also called Bayesian networks}
  \]

- Can infer conditional independencies from graphical model alone!
Graphical model for medical diagnosis

This model makes several assumptions:
1. \(d_i \perp d_j\)
2. \(f_i \perp f_j \mid d\)

Joint distribution factors as:

\[
P(f, d) = P(f \mid d)P(d) = \prod_i P(f_i \mid d) \prod_j P(d_j)
\]

"Noisy or" distribution

Prior probability of having disease

Marginal inference:

\[
Pr(d_i \mid f)
\]

MAP inference:

\[
\arg \max_d Pr(d \mid f)
\]

Having a probabilistic model allows us to quantify our uncertainty

How does one learn the model?

(Miller et al., ‘86, Shwe et al., ‘91)
Learning

• Suppose we had historical data \(\{(x, y)^1, \ldots, (x, y)^l\}\)
  • Assume drawn from the true distribution \(\Pr(x, y)\)
  • Complete data (no variables unobserved)

• Find the parameters of the model that maximize the likelihood of the data, \(\prod_l \Pr(x^l, y^l; \theta)\)

• In directed graphical models, ML estimation from complete data is easy -- simply calculate statistics

How many parameters?
Inference

• Recall, to find out where the missile is now, we do marginal inference:
  \[ \Pr(x_n \mid y_1, \ldots, y_n) \]

• How does one compute this?

• Applying Bayes’ rule, we reduce to computing
  \[ \Pr(x_n \mid y_1, \ldots, y_n) = \frac{\Pr(x_n, y_1, \ldots, y_n)}{\Pr(y_1, \ldots, y_n)} \]

• Naively, would seem to require \( k^{n-1} \) summations,
  \[ \Pr(x_n, y_1, \ldots, y_n) = \sum_{x_1, \ldots, x_{n-1}} \Pr(x_1, \ldots, x_n, y_1, \ldots, y_n) \]
Marginal inference in HMMs

- **Use dynamic programming**

\[
\Pr(x_n, y_1, \ldots, y_n) = \sum_{x_{n-1}} \Pr(x_{n-1}, x_n, y_1, \ldots, y_n)
\]

\[
= \sum_{x_{n-1}} \Pr(x_{n-1}, y_1, \ldots, y_{n-1}) \Pr(x_n, y_n \mid x_{n-1}, y_1, \ldots, y_{n-1})
\]

\[
= \sum_{x_{n-1}} \Pr(x_{n-1}, y_1, \ldots, y_{n-1}) \Pr(x_n \mid x_{n-1}) \Pr(y_n \mid x_n, x_{n-1})
\]

\[
= \sum_{x_{n-1}} \Pr(x_{n-1}, y_1, \ldots, y_{n-1}) \Pr(x_n \mid x_{n-1}) \Pr(y_n \mid x_n)
\]

- **For n=1, initialize** \( \Pr(x_1, y_1) = \Pr(x_1) \Pr(y_1 \mid x_1) \)

- **Total running time is \( O(nk) \) – linear time!** **Easy to do filtering**
MAP inference in HMMs

- MAP inference in HMMs can also be solved in linear time!

\[
\arg \max_x \Pr(x_1, \ldots, x_n \mid y_1, \ldots, y_n) = \arg \max_x \Pr(x_1, \ldots, x_n, y_1, \ldots, y_n) \\
= \arg \max_x \log \Pr(x_1, \ldots, x_n, y_1, \ldots, y_n) \\
= \arg \max_x \log \left[ \Pr(x_1) \Pr(y_1 \mid x_1) \right] + \sum_{i=2}^{n} \log \left[ \Pr(x_i \mid x_{i-1}) \Pr(y_i \mid x_i) \right]
\]

- Formulate as a shortest paths problem

Weight for edge \((s, x_1)\) is
\[
\log \left[ \Pr(x_1) \Pr(y_1 \mid x_1) \right]
\]
Weight for edge \((x_{i-1}, x_i)\) is
\[
\log \left[ \Pr(x_i \mid x_{i-1}) \Pr(y_i \mid x_i) \right]
\]
Weight for edge \((x_n, t)\) is 0

Called the Viterbi algorithm
Applications of HMMs

• Speech recognition
  – Predict phonemes from the sounds forming words (i.e., the actual signals)

• Natural language processing
  – Predict parts of speech (verb, noun, determiner, etc.) from the words in a sentence

• Computational biology
  – Predict intron/exon regions from DNA
  – Predict protein structure from DNA (locally)

• And many many more!
How to generalize?

- How do we do inference in these models?

NP-hard

Linear time
Undirected graphical models

- **Markov random fields** provide an alternative parameterization of joint distributions, corresponding to an *undirected* graph

Pairwise model:  
\[ \Pr(x) = \frac{1}{Z} \prod_{i,j \in E} \psi_{ij}(x_i, x_j) \]

Partition function (normalization constant)
\[ Z = \sum_x \prod_{i,j \in E} \psi_{ij}(x_i, x_j) \]

Non-negative function of two variables

<table>
<thead>
<tr>
<th></th>
<th>(X_j = 0)</th>
<th>(X_j = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_i = 0)</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>(X_i = 1)</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

- Just as before, graphical model implies *conditional independence* properties, e.g.  
  \[ X_1 \perp X_4 \mid X_3 \]
HMM as an undirected model

\[ \psi(x_1, x_2) = \Pr(x_1) \Pr(x_2 \mid x_1) \Pr(y_1 \mid x_1) \Pr(y_2 \mid x_2) \]
\[ \psi(x_i, x_{i+1}) = \Pr(x_{i+1} \mid x_i) \Pr(y_{i+1} \mid x_{i+1}) \text{ for } i = 2, \ldots, n - 1 \]

Next, we generalize the dynamic programming algorithm used for inference in HMMs to inference in tree-structured MRFs
Belief propagation

1. Fix a root

2. Pass messages from the leaves to the root

3. Pass messages from the root to the leaves
The messages are always of the form:

\[ m_{i \rightarrow j}(x_j) = \sum_{x_i} \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i) \]

- \( m_{4 \rightarrow 3}(x_3) = \sum_{x_4} \psi_{3,4}(x_3, x_4) \)
- \( m_{3 \rightarrow 2}(x_2) = \sum_{x_3} \psi_{2,3}(x_2, x_3)m_{4 \rightarrow 3}(x_3) \)
- \( m_{5 \rightarrow 2}(x_2) = \sum_{x_5} \psi_{2,5}(x_2, x_5) \)
- \( m_{1 \rightarrow 2}(x_2) = \sum_{x_1} \psi_{2,1}(x_2, x_1) \)
Sum-product belief propagation

The messages are always of the form:

\[ m_{i \rightarrow j}(x_j) = \sum_{x_i} \psi_{i,j}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i) \]

Step 3

\[ m_{2 \rightarrow 1}(x_1) = \sum_{x_2} \psi_{2,1}(x_2, x_1) m_{5 \rightarrow 2}(x_2) m_{3 \rightarrow 2}(x_2) \]

Step 4

\[ \Pr(x_i) = \frac{\prod_{j \in N(i)} m_{j \rightarrow i}(x_i)}{\sum_{\hat{x}_i} \prod_{j \in N(i)} m_{j \rightarrow i}(\hat{x}_i)} \]

Applied to the HMM, this would compute \( \Pr(x_i \mid y_1, \ldots, y_n) \) for all i
Max-product belief propagation

The messages are always of the form:

\[ m_{i \rightarrow j}(x_j) = \max_{x_i} \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i) \]

When the MAP assignment is unique, local decoding succeeds in finding it:

\[ x_i^{\text{MAP}} = \arg \max_{\hat{x}_i} \prod_{j \in N(i)} m_{j \rightarrow i}(\hat{x}_i) \]
The need for approximate inference (Example: stereo vision)

- How far away are the objects in the images?

**Undirected graphical model**

$$G = (V, E)$$

$$\theta_{ij}(x_i, x_j) = \log \psi_{ij}(x_i, x_j)$$

Even for a small 100x100 pixel image, model has 10,000 variables

**Inference**

$$\arg\max_x \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j)$$
Approximate inference

• Two broad classes of approximate inference algorithms are:
  – Monte-carlo methods (e.g., likelihood weighting, MCMC)
  – Variational methods

• Popular variational method is **loopy belief propagation**
  – Initialize messages in BP to 1
  – Run BP algorithm until convergence, iteratively choosing a new edge to send a message along
  – Few guarantees of correctness. May not even converge!

• Much progress has been made in the last 15 years on approximate inference algorithms
Conclusion

• Graphical models are a powerful framework
  – Large number of real-world problems can be formulated as graphical models
  – Allows us to explicitly model uncertainty in predictions

• Key problems to solve are learning and inference

• Hidden Markov models have efficient inference algorithms based on dynamic programming
  – Algorithms are called forward-backward (marginal inference) and Viterbi (MAP inference)

• Message-passing algorithms allow efficient exact inference in any tree-structured Markov random field
  – Can use as an approximate inference algorithm in graphs with loops

• Exciting field at the intersection of optimization, statistics, and algorithms
Want to learn more?

Next semester I am teaching
CSCI-GA.3033-006:
“Special Topics in Machine Learning:
Probabilistic Graphical Models”